Collection Performance Computation

25-April-2001

Outline

- Introduction
- References
- Stellar Distribution
- Star Processing Flow
- Processing Requirement
- Processor Capability
- Performance Prototype
- Summary

Introduction

- **Mission figure of merit**
  - **Number of stars imaged during mission**
- **A driving requirement for the FAME instrument is the number of stars per second that can be collected.**
- **Potential bottlenecks**
  - **Downlink capacity**
    - **Bandwith: 300 kb/s**
    - **Number of bits per star**
  - **Flight CPU performance**
    - **Window determination**
    - **CCD command generation**
- **This presentation considers on-board computation**

Introduction (2)

- **Design drivers**
  - ➨ **Maximum stellar density per update period**
  - **Efficiency of accessing star catalog**
  - **Computation requirements per star**
  - **CPU performance**
  - **Star collection packetizing overhead**

References

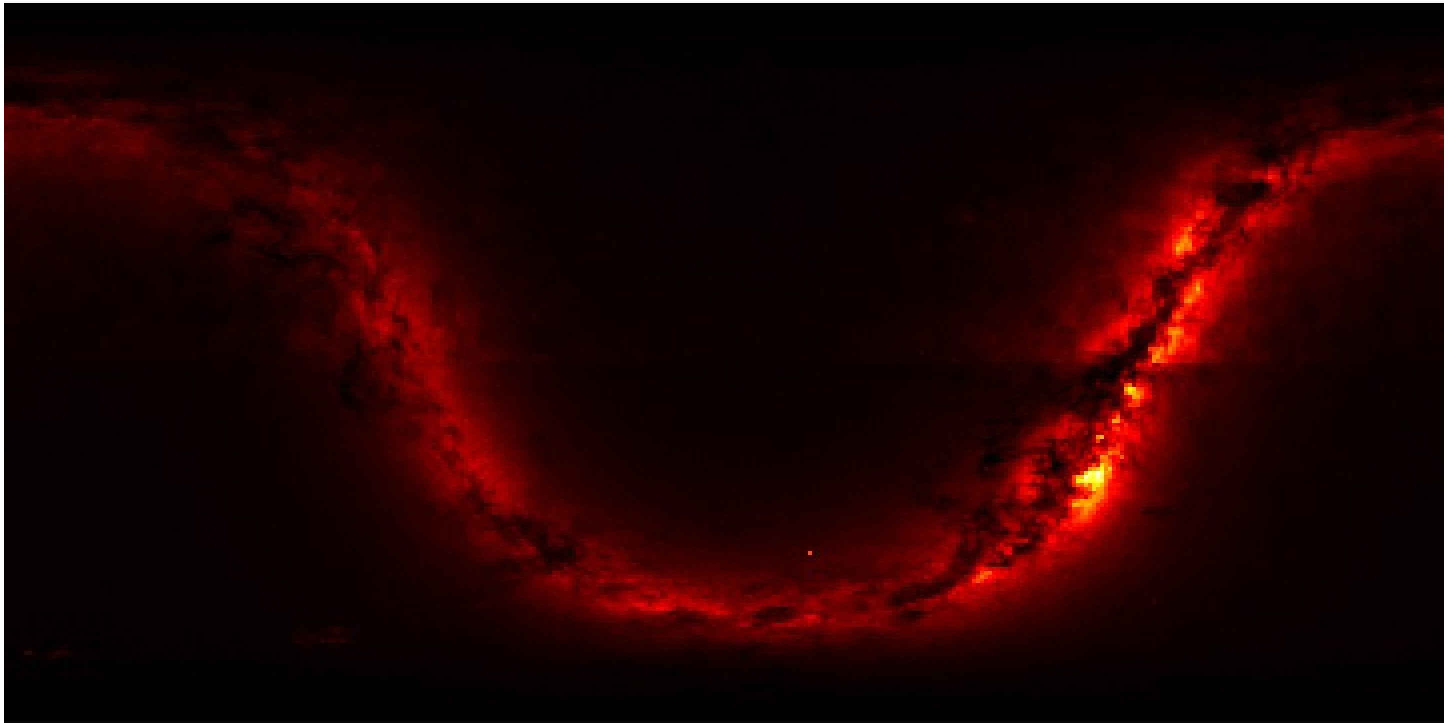
- **Gaume**
- **Kaplan**
- **Monet**

Stellar Distribution

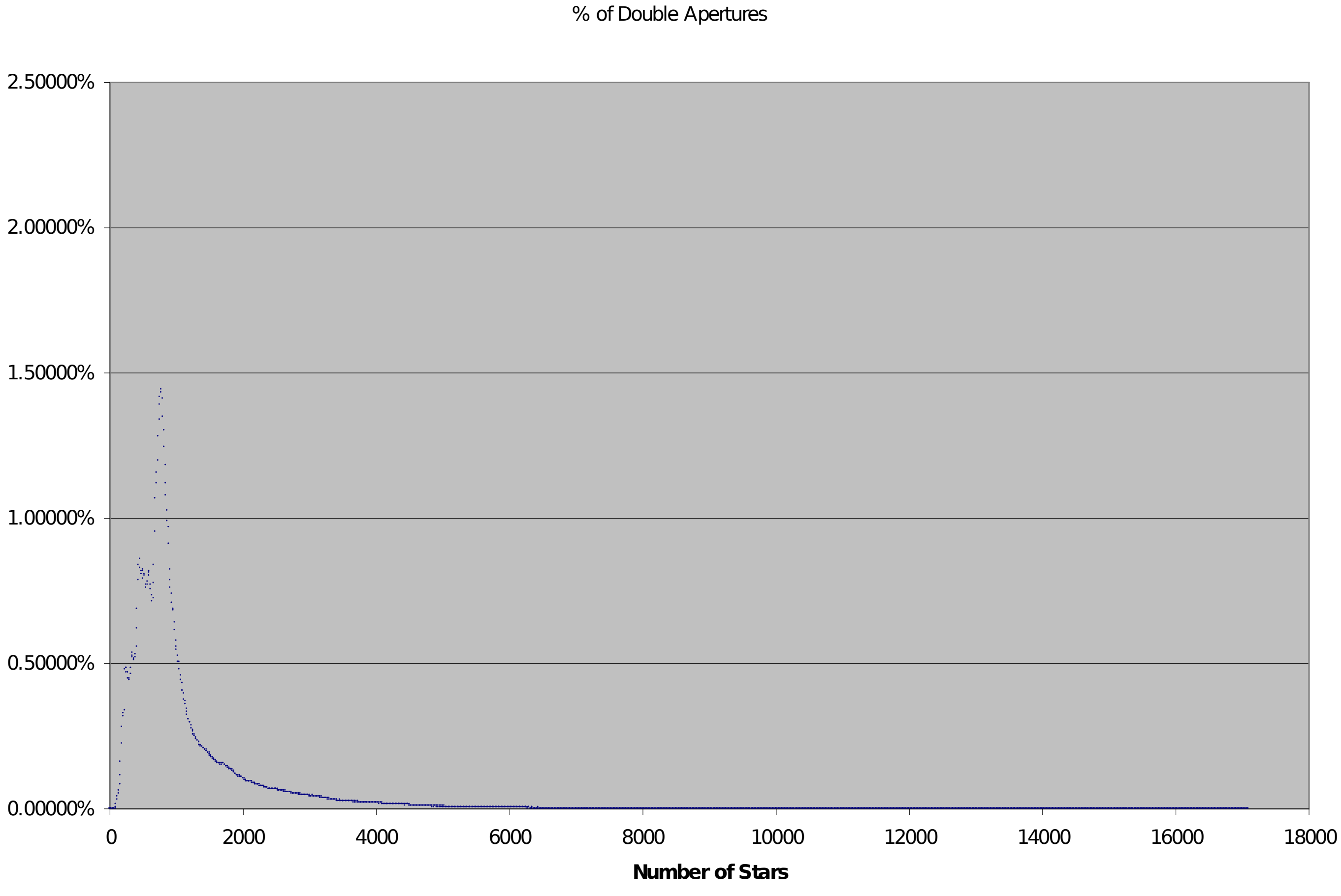
Stellar Density

- **Average number of stars per second is ??**
- **Maximum number of stars per second is ??**
- **Over sufficiently long periods the number stars in the FOV will tend to average number of**
  - **Non-galactic roll periods to rapidly tend to the average**

Stellar Distribution

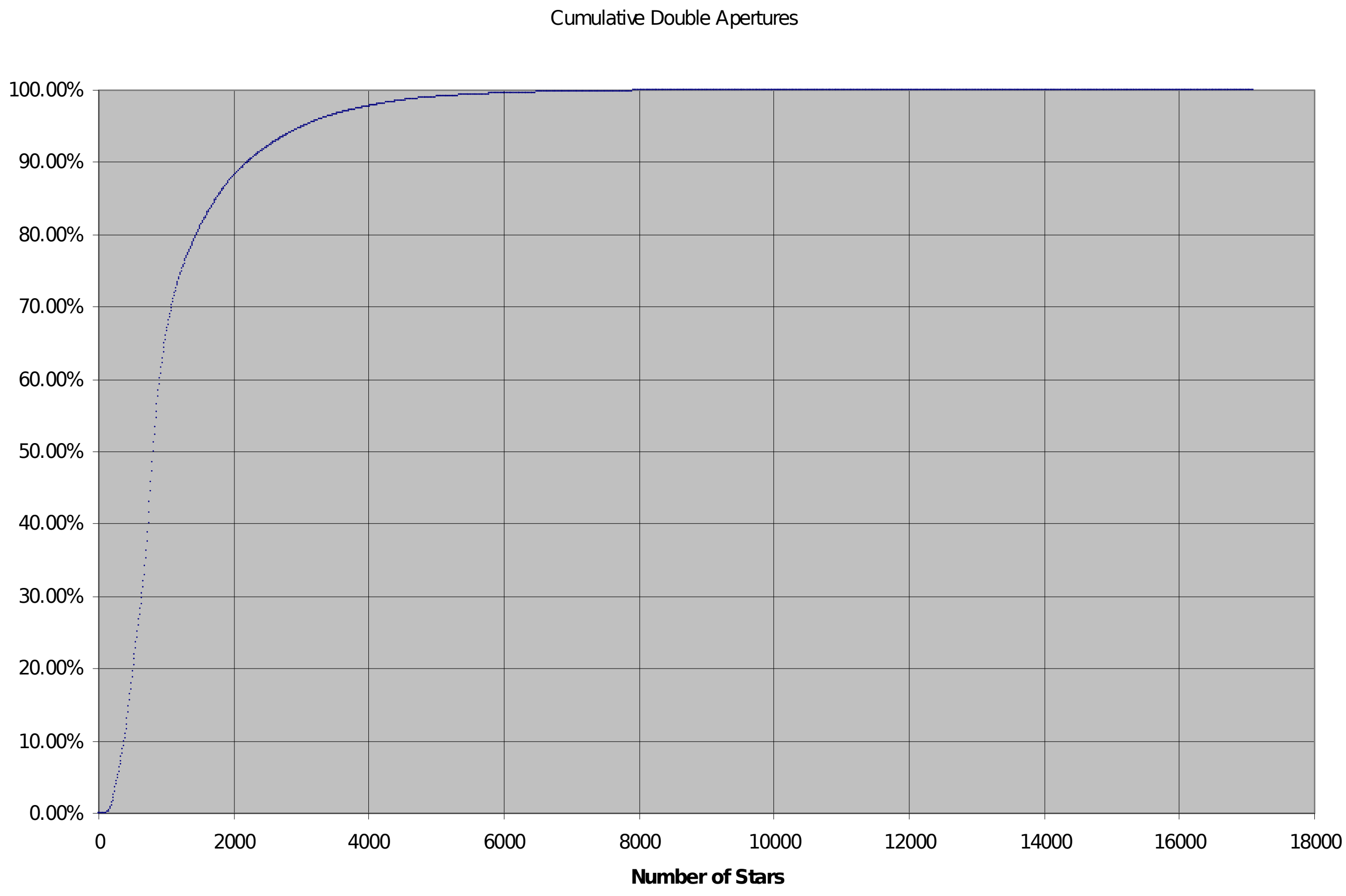


Stars in FOV



% of Double Apertures
2.50000%
2.00000%
1.50000%
1.00000%
0.50000%
0.00000%
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Number of Stars

Cumulative Numbers

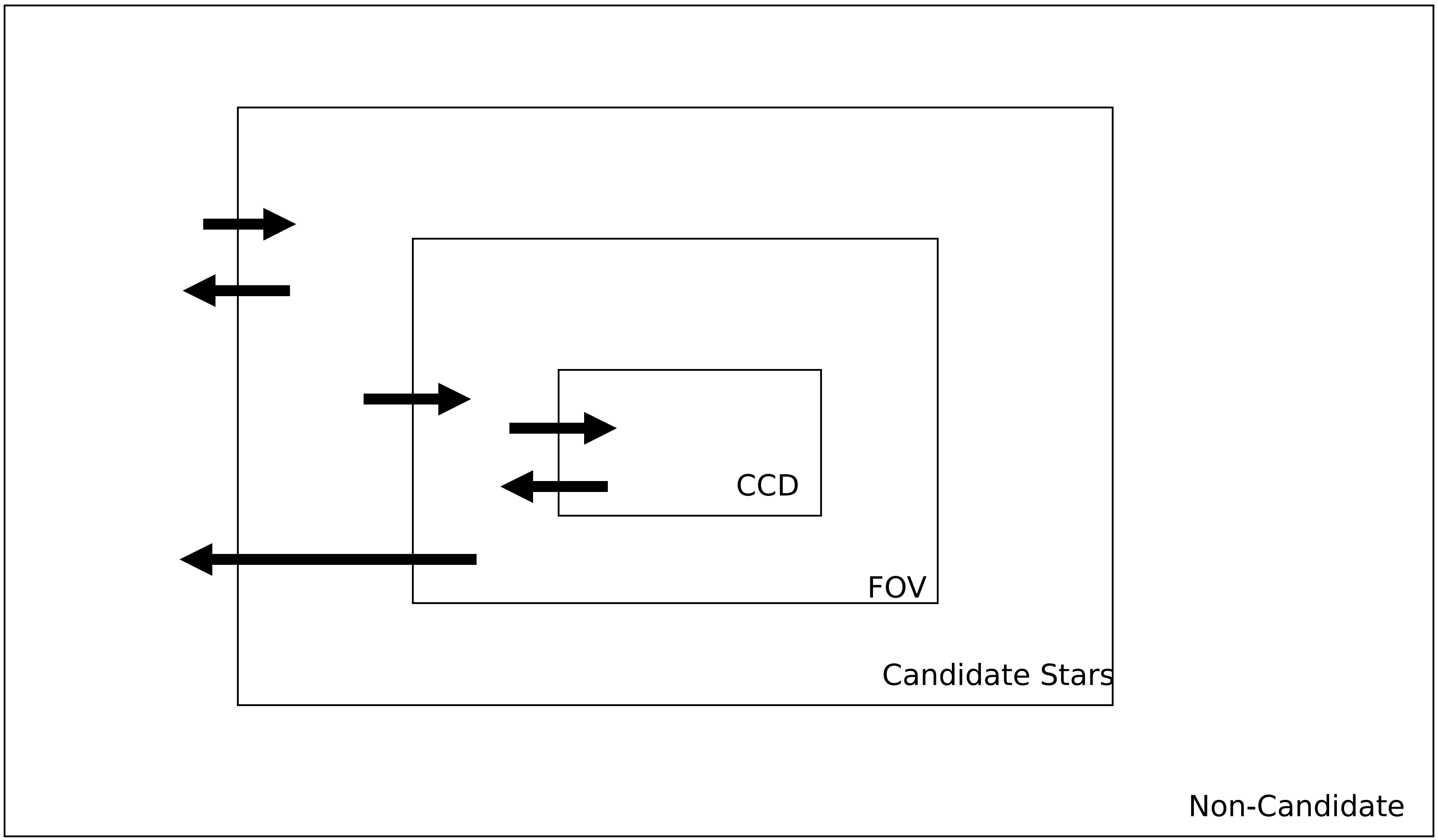


Analysis of Star Processing

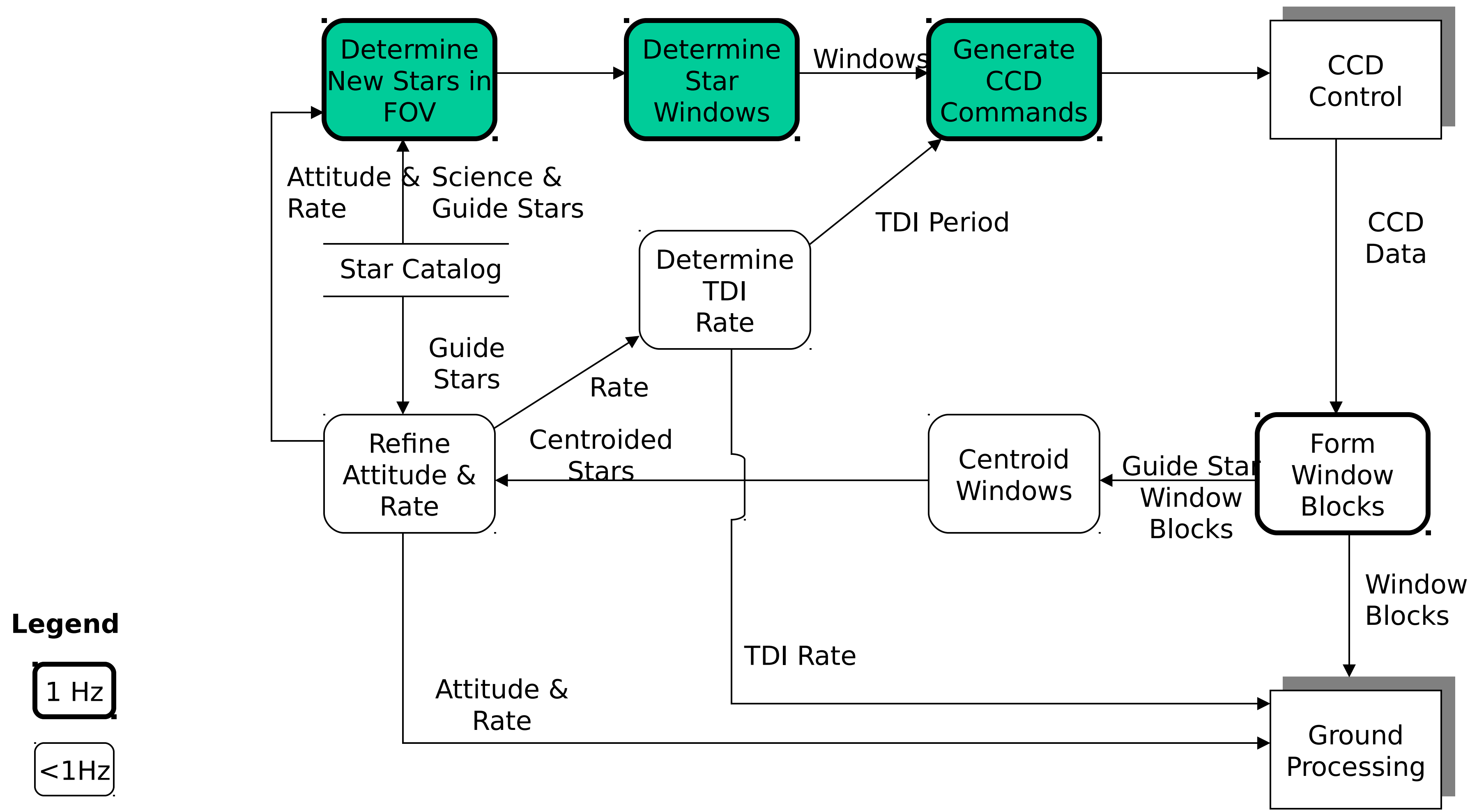
Star States

- **Non-candidate Star**
  - **In catalog but not near current FOV in current period**
- **Candidate Star**
  - **Close to field of view during current period**
- **Field of View (FOV)**
  - **In FOV during current period**
- **CCD**
  - **On a CCD during current period**

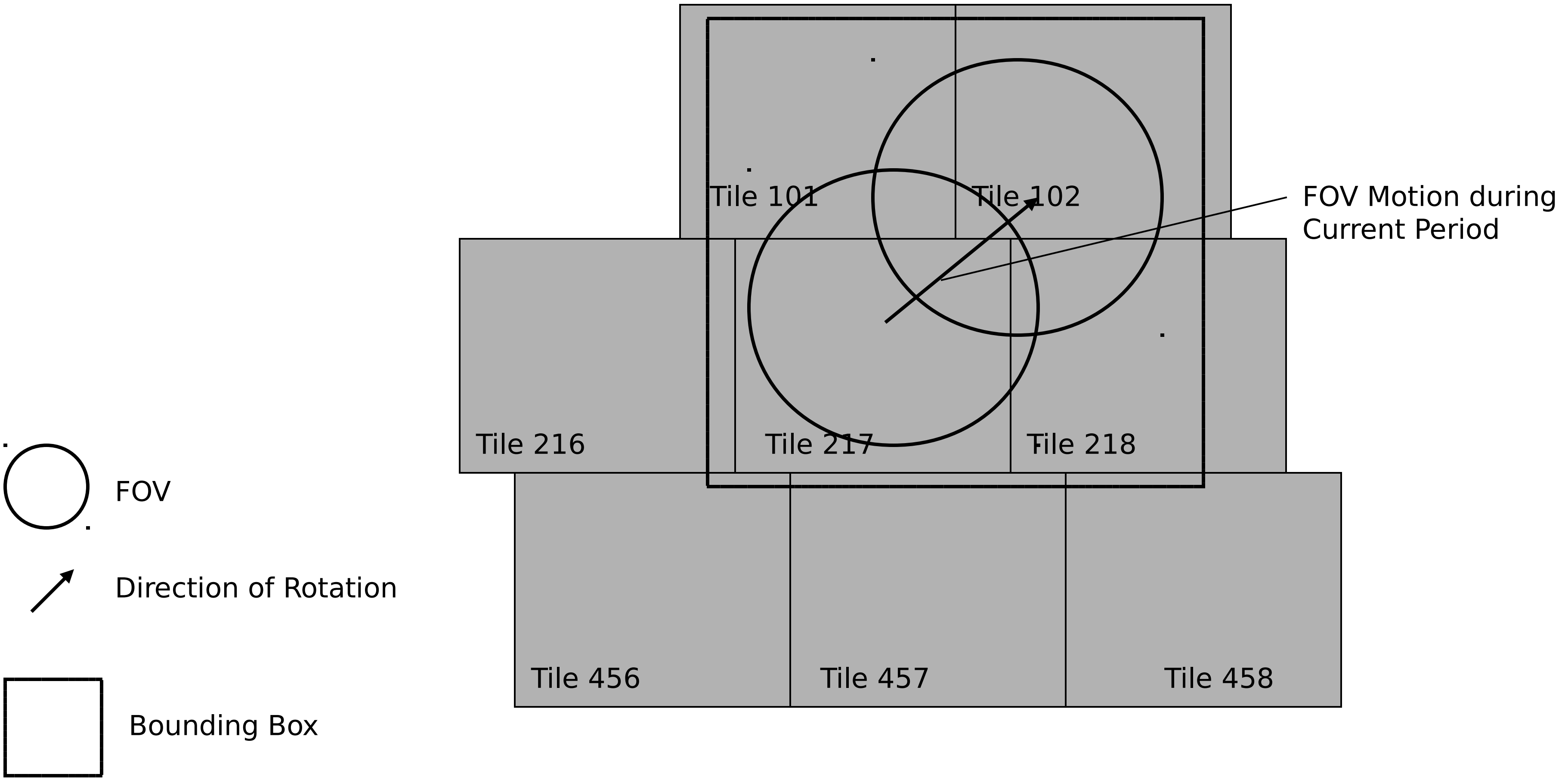
Star States



Science Mode Dataflow



Determine Candidate Stars



Determine Candidate Stars

Method 1:

```
For each Aperture
        For each subdivision of the Update Period
        {
                Generate aperture vector
                For each tile
                        If (tile center vector is close to aperture vector)
                                Add stars in tile to candidate star list
        }
```

Method 2:

```
For each Aperture
        Form Bounding Box around final aperture of last period and final aperture
        of current period
        For each tile
                If (tile boundaries in bounding box)
                        Add stars in tile to candidate star list
```

Determine Stars in FOV & CCD

```
For every update period
{
    Update FAME Attitude
    Update FAME Rate
    Compute Aperture vectors
    Get Candidate Stars
    Compute Tile Aberration
    For each candidate star
    {
        Correct for Parallax
        Correct for Stellar Aberration
        Project on to Focal Plane
        For each CCD
        {
            Compute Crossing time & position
        }
    }
}
```

Generate CCD Commands

```
For CCD
        For each star with crossing time
        {
                Form charge injection command
                If no window in keep out zone
                         Add command to command queue
                For star window
                {
                        Form CCD read commands
                        Add commands to command queue
                }
                If previous charge injection keep out zone overlaps window
                        Remove charge injection command
        }
}
```

Processing Requirement

Scalar Computations

- **CCD commands**
  - **Estimated 100 OPS per command**
  - **13 commands per window**
- **Star Retrieval**
  - **Estimated 100 OPS per star**
- **Construction Guide Star Window**
  - **2 guide stars/sec**
  - **200 pixels/star**
  - **10 OPS/pixel**
  - **4,000 OPS**

Floating Pt Computations

Computation	Equation	Fmult	Fadds	Fsqrt	Fother	Fmult/s	Fadd/s	Fsqrt/s	Fother/s	Hz
Update **F**	**F**+**F'**.dt	3	3			3	3	0	0	1
Update **x(t)**, **y(t)**, **z(t)**	**x(t)**=**x(t0)**+**W**.dt;**y(t)**=**y(t0)**+**W**.dt;etc	9	9			9	9	0	0	1
Update **q**	**-x(t)**sin(phi/2)+**y(t)**cos(phi/2)	6	3			12	6	0	0	2
Total per update						**24**	**18**	**0**	**0**	
Select Tile	**p.q**>0.999759	3	2		1	18	12	0	6	6
Compute tile aberration	||**B**+**F'**/c||	6	5	1		18	15	3	0	3
Total (Tiles)						**36**	**27**	**3**	**6**	
Compute parallax	**p-F**		3			0	6	0	0	2
Compute star aberration	**p+A**		3			0	6,000	0	0	2,000
Project star on focal plane	**p.u, p.v**	6	4			12,000	8,000	0	0	2,000
Determine pixel crossing time	t+**u**.(**h-s**)/w	4	6			8,000	12,000	0	0	2,000
Total (Stars)						**20,000**	**26,006**	**0**	**0**	
Grand Total	**46,120**					20,060	26,051	3	6	
Legend		#dims				**Parameters**				
Fame position vector	**F**	3				Number stars/period	2,000			
Fame velocity vector	**F'**	3				% Parallax stars	0.10%			
Delta time	dt					Candidate tiles	6			
Star position in (x,y,z) [ICRS]	**p**	3				%Selected Tiles	50%			
Spacecraft orientation	**x(t),y(t),z(t)**	3*3								
Aperture in (x,y,z) [ICRS]	**q**	3								
Velocity of light	c									
Aperture angle	phi									
Abberation	**A**	3								
Box center vector	**B**	3								
Instrument coordinates	**q, u, v**									
Spacecraft coordinates	**x, y, z**									
Star position in (u,v)	**p'**	2								
Pixel coordinates (u,v)	**h**	2								
Rotation velocity	w									

Processor Capability

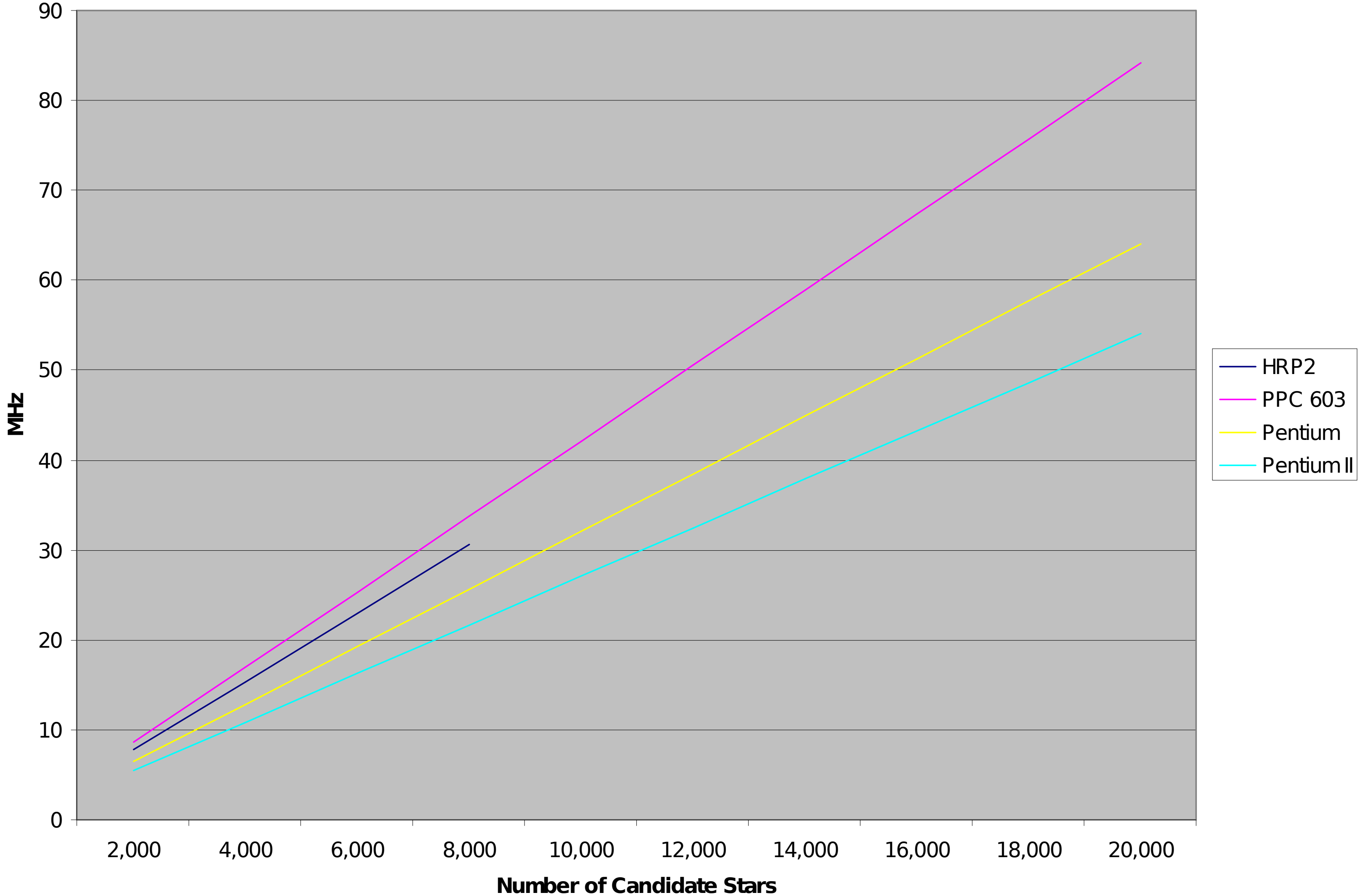
CPU Performance

- **Integer operations**
  - **Approximately one clock cycle per operations**
- **Floating point operations**
  - **Multiple cycles per operation (often 5-20)**
  - **Sometimes have add/multiply operation (I.e. faster dot products)**
  - **Older processors require more cycles for double precision**
- **Dependent on memory organization**
  - **Cache levels and size**
- **Best test of performance is a prototype of the kernel code on an actual processor**

CPU Performance

- **Assume 300% Margin**
- **Assume double precision**

Processor vs Performance



Star Processing Prototype

Activity

- **Develop a prototype of all the algorithms used to perform star processing**
  - **C/C++ developed on PC**
- **Time performance on various processors**
- **Scale performance to selected processor**

Status

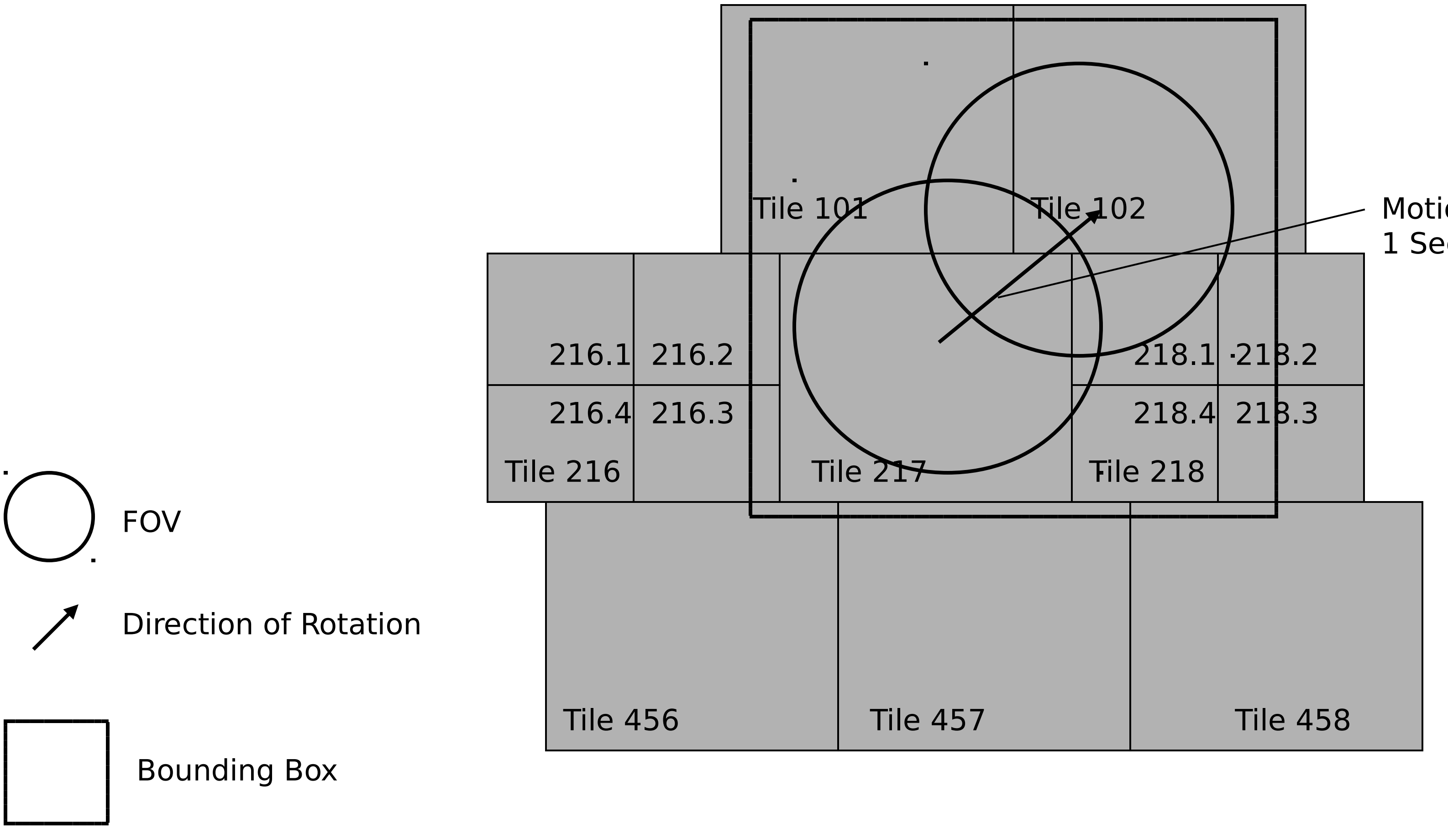
- **Window formation is nearly complete**
  - **Preliminary results at IPDR**
- **Catalog management under way**

Summary

- **Science requirement drives the selection and design of the software**
- **Estimated performance requires a 40+ MHz processor to provide margin**

Backup

Use of Quad-Trees



Floating Pt Test Program

```
/* Floating Addition */

for (i=0;i<ARRAY_SIZE; i++)
        a[i]=b[i] =i;

start=clock();
for (j=0; j<ITERATIONS; j++){
        for (i=0;i<ARRAY_SIZE; i++)
                a[i]=a[i]+b[i];
};
end=clock();

cpu_add = ((float)(end-start)/CLOCKS_PER_SEC);
adds = ((ARRAY_SIZE*ITERATIONS)/cpu_add)/1e6;
cout << "Addition loop " << cpu_add << " CPU seconds" << endl;

/* Floating Multiplication */

for (i=0;i<ARRAY_SIZE; i++)
        a[i]=b[i]=i;

start=clock();
for (j=0; j<ITERATIONS; j++){
        for (i=0;i<ARRAY_SIZE; i++)
                a[i]=b[i]*b[i];
};
end=clock();

cpu_mult = ((float)(end-start)/CLOCKS_PER_SEC);
mults = ((ARRAY_SIZE*ITERATIONS)/cpu_mult)/1e6;
cout << "Multiplication loop " << cpu_mult << " CPU seconds" << endl;

/* Dot Product */

for (i=0;i<ARRAY_SIZE; i++)
        a[i]=b[i]=i;
        a_dot_b = 0;

start=clock();
for (j=0; j<ITERATIONS; j++){
        for (i=0;i<ARRAY_SIZE; i++)
                a[i]=b[i];
                a_dot_b+= a[i]*b[i];
};
end=clock();
```

Processor Performance (Mops)

	Fadd	Fmult	Fdot	DFadd	DFmult	DFdot
PowerPC 603 75MHz	3.6	3.6	9.4	3.6	3.4	9.4
PowerPC 601 75MHz	4.6	4.6	10.6	4.7	4.7	10.7
PowerPC 601 100MHz	6.2	6.2	14.2	6.3	6.3	14.3
Pentium 233Mhz MMX	27.4	21.3	38.2	27.4	19.6	21.1
Pentium II 266Mhz	50.5	50.5	181.8	62.9	42.3	172.4
Pentium II 300MHz	58.7	58.7	148.9	72.3	48.5	196.85